02

Idempotência

Nós vimos que o verbo, o método, GET é usado para requisitar ou pedir a representação de um recurso. E o método POST é usado para criar um recurso no servidor. Ambos trabalham com a representação do recurso, o GET recebe ela enquanto o POST envia. Na resposta do POST você pode ter 201 Created ou ainda um 200 ok com o conteúdo da representação.

Mas qual uma outra grande diferença entre eles? Se o GET só pede dados, isso significa que ele não altera nada no servidor. Eu posso fazer a mesma requisição duas vezes, três vezes, cinco vezes, cem vezes, um milhão de vezes, que não vou alterar nada no servidor.

Chamamos esse tipo de requisição de idempotente, ela pode ser executada quantas vezes for, e o resultado não será alterado. O GET é um tipo de requisição idempotente. Claro que se entre uma requisição GET e outra alguém remove um produto, o produto não aparecerá, mas isso não é resultado da sua requisição GET: enquanto você faz requisições GET, requisições idempotentes, o resultado da execução não é acumulado, ele é executado uma única vez. Claro, você pode ter contadores ou algo do gênero, mas o importante é que a tarefa principal do GET não execute algo que acumula, que altera o estado do servidor a cada nova requisição.

Algo muito legal de requisições idempotentes é que se a conexão cai no meio da requisição, será que o servidor recebeu a requisição ou não? Você não tem como saber, mas não importa, basta executar a requisição novamente. Não tem problema em executá-la novamente. Requisições idempotentes são poderosas por terem essa tolerância a falha naturalmente.

As requisições POST não são idempotentes, se você executar a mesma requisição duas vezes, você alterará o estado do servidor novamente, possivelmente executando a tarefa duas vezes, em nosso caso criando dois carrinhos. Portanto, sempre que possível, extraímos dados somente com GET. E sempre que precisamos alterar algo, temos que usar POST (não podemos usar GET, pelo risco de acessar a mesma URI diversas vezes e alterar os dados diversas vezes).

O POST cria um conteúdo, cria um novo recurso, não é idempotente. Se acessá-lo duas vezes, você corre o risco de ficar com dois recursos iguais no servidor. A escolha é simples: buscar dados, GET idempotente, criar algo POST não idempotente.